



ITeV

Docket No.: 10003507-2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of:

Kenneth W. SHRUM et al.

Serial No. 09/825,403

Group Art Unit: 2172

Confirmation No. 1629

Filed: April 3, 2001

Examiner: Hung Q. Pham

For: INTERNET SERVER SYSTEM TEST AND MEASUREMENT

**RESPONSE TO EXAMINER REQUEST FOR INFORMATION**

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Sir:

This is in further response to the Office Action mailed on June 17, 2004, for which a corresponding response was filed on September 17, 2004.

Enclosed is the Examiner requested HP Firehunter Concept Guide.

If there are any formal matters remaining after this response, the Examiner is requested to telephone the undersigned to attend to these matters.

Respectfully submitted,

Date:

10/20/04

By:

Stephen T. Boughner  
Registration No. 45,317



**HP Firehunter**

---

Internet Service Management Solutions

## Concept Guide



**Congratulations on your purchase of HP Firehunter!** As an Internet service provider, you know by now that the only sure way to grow market share and glean profits from revenues is to offer your customers unique value-added services and to back those services with comprehensive quality guarantees. Firehunter is a highly customizable and scalable Service Management solution that enables ISPs and Enterprise/IT providers to deliver managed value-added services with assured and verified quality. Firehunter can automatically measure, monitor and verify “carrier-grade” quality for the Internet services you offer. It provides the service assurance and reporting capabilities needed to differentiate your service offerings with SLA guarantees, starting with mail, news and web and expanding in the future to a full suite of value-added services such as Web-Hosting, Virtual Private Network (VPN), and E-Commerce. With Firehunter you can distance yourself from the pack and take a leadership position in this new high-growth e-business market. The product offers the following unique capabilities:

- **Service Level Agreement (SLA) operational monitoring and reporting that enable you to manage your SLAs.** Firehunter allows you to define SLAs for individual customers based on any set of measurements in your service model, such as response time, availability, and so on. It allows you to monitor service-level performance in real-time and to identify and correct problems before SLAs are breached. You can automatically deliver “SLA report cards” to customers to verify SLA compliance and improve customer satisfaction. You can also use this feature to investigate and validate new SLAs before you begin offering them.
- **A unique service modeling capability that enables a top-down view of service health and rapid root cause analysis of problems.** Firehunter’s service model enables top-down visualization of all the components of the service delivery that are relevant to quality of service (QoS). Firehunter’s service model goes beyond the simple collection of data and generation of alerts offered by other products, to capture aggregate and correlate service performance metrics to provide useful information, not just raw data. It automates the analysis that is too often done in the heads of a small number of system experts. Firehunter enables you to visualize your complete service environment and the performance

of key components end-to-end. It allows you to quickly navigate to the source of a problem – even in the most complex environments.

- **Customizable service models, measurements and reports that enable end-to-end management of ISP network, server, and application components.** Active measurements of your services, such as web, email, and news, enable you to assess the quality of service you offer to your subscribers. Performance and usage measurements of infrastructure services such as the Domain Name Service (DNS) facilitate problem diagnosis. Measurements of server and network performance and throughput complete a comprehensive measurement suite. ISPs can easily tailor Firehunter to their specific environment, needs, and processes. They can add and configure measurements, thresholds, and status propagation rules. They can also set test frequency, configure measurement graphs and define service-level reports. HP consultants can also add new service models and discovery engines to customize the Firehunter/PRO product for new services.
- **Automated Reporting.** Firehunter's reporting capabilities enable ISPs to define, and schedule for automatic delivery, custom reports to internal staff and customers. This capability substantially reduces the manual effort required by most ISPs with current management systems and procedures. Improved timeliness and accuracy of reports results in increased customer satisfaction. Historical data storage enables a variety of reports on services managed and associated SLAs.
- **A sophisticated set of graphical interfaces that visually alert you to problems in your system and assist you in doing your job.** The GUI includes windows specifically designed for help desk personnel, operators, and system planners, as well as interfaces to make configuration and customization as painless as possible. Multiple remote GUIs can be configured and can operate simultaneously to provide dedicated consoles for management system users.
- **Web-enabled management.** Firehunter lets you export events, annotations, graphs, planning reports and SLA "report cards" so that they are viewable anywhere from a browser. You can check

service status without being at the Firehunter console. Your help desk personnel can see service-related status information when customers call in, and your customers can verify they are getting the service they want, all through the web.

- **The ability to manage by exception, quickly analyzing and intelligently responding to any deviations from “normal” behaviors.** Firehunter automatically computes a (normal performance) baseline for all key service measurements. Thresholds can be configured relative to baselines to provide automatic detection of service anomalies. An Action Manager allows configuration of automatic actions such as pager alerts, email notifications, and scripts, to be generated when anomalies occur.
- **Easy integration with other tools you are already using.** We realize you’ve already made quite an investment in monitoring tools. Firehunter doesn’t squander that investment, but puts it to work, it enables you to launch and use existing tools from within the service model being managed or diagnosed. You need not lose or duplicate data that you are already gathering, but can incorporate it into the Firehunter console to compare with other data for a more complete picture of your system performance.
- **Scalability and Adaptability.** Scalability for large environments is achieved with Firehunter/PRO through hierarchical distribution of Diagnostic Measurement Servers (DMS) and measurement agents. The servers are integrated to form a composite Service Model representing the entire service environment. The Service Model may be partitioned and distributed to support very large (Global) Internet environments. Multiple computing platforms are supported for flexibility and adaptability to your service infrastructure. The Firehunter DMS is presently supported on Windows NT®, Solaris®, and HP-UX. Measurement agents are available for all leading PC and UNIX® platforms typically found in ISP environments.

This Concepts Guide provides background information about these features and guidelines for their use. It includes real-life stories of how some ISPs have put the power of Firehunter to work. We invite you to

use these ideas, expand on them, and invent your own as you realize the full potential of Firehunter!

Windows NT® is a US registered trademark of Microsoft Corporation, Solaris® is a US registered trademark of Sun Microsystems, Inc., UNIX is a registered trademark of the Open Group







---

## Concept Guide Contents

### Chapter

Preface

System Architecture\_\_\_\_\_1

Service Model and Service Health\_\_\_\_2

Tests and Measurements\_\_\_\_\_3

Thresholds and Baselines\_\_\_\_\_4

Actions\_\_\_\_\_5

Integration with Other Tools\_\_\_\_\_6

SLAs and Reporting\_\_\_\_\_7

License and Warranty\_\_\_\_\_A

## System Architecture

The Firehunter product, and the architecture upon which it is based, include two key enabling technologies: baselining and variable thresholds, and service modeling. Without these technologies Firehunter would not be able to monitor and correlate the vast quantities of data sampled from the ISP environment.

The Firehunter architecture consists of the following three tiers:

- Graphical user interface (GUI) clients
- Diagnostic measurement servers (DMS)
- Agents

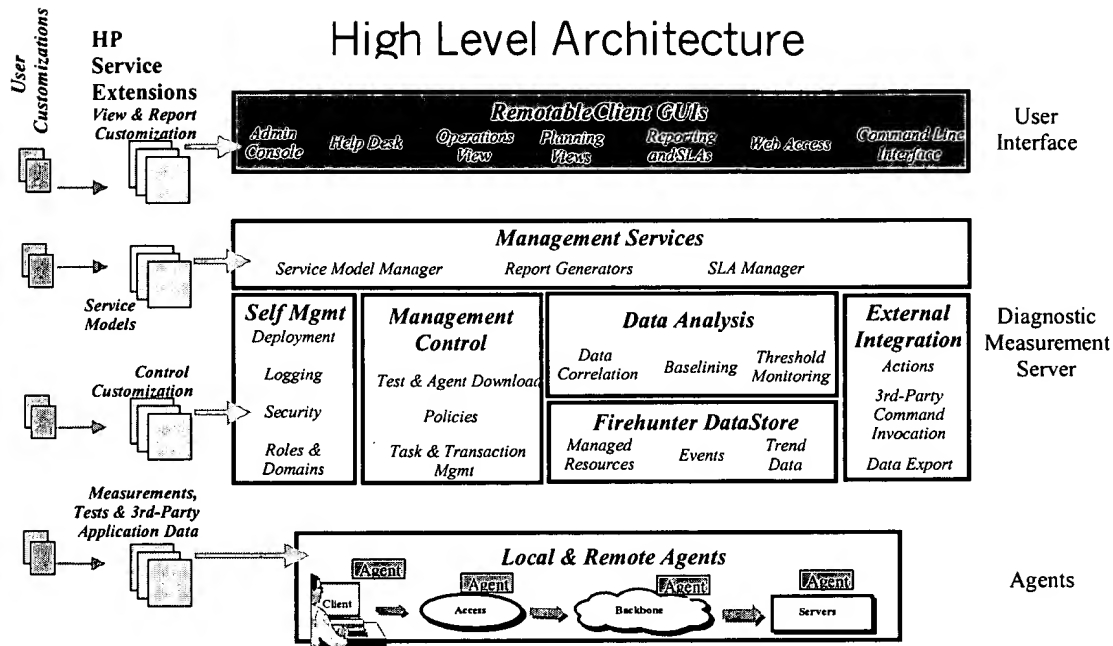
In the most basic configuration, you'll run only one DMS, with its own, local agent. As you grow or your needs change, you can expand this configuration by adding remote agents and in Firehunter/PRO DMSs. This chapter describes the basic architecture and its scalability.

---

## Basic architecture

Below is a diagram of the Firehunter architecture. It shows the three tiers of the architecture: the User Interface, the Diagnostic Measurement Server, and the Agent, each of which is explained below.

## HP Firehunter High Level Architecture



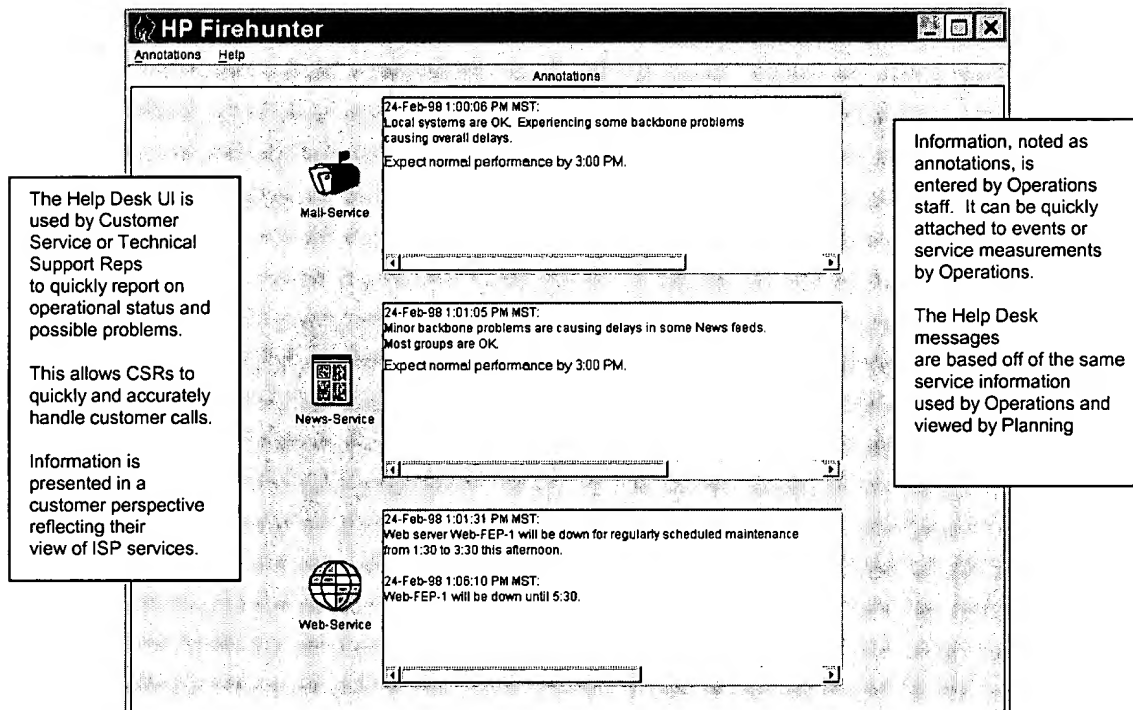
### Client GUI

The Client GUI gives you access to the status summaries and detailed information regarding the quality of the services you are providing. The GUI enables you to configure Firehunter and monitor the health of your services as well as view events, measurement data, and reports. For each service, you can navigate its dependent resources, their status, and the graphical data being gathered to monitor them.

The GUI is accessed as an application, installed with the DMS or installed separately on a remote machine, so you can run one or more GUI clients local to the DMS or remote from it.

The GUI consists of the following windows:

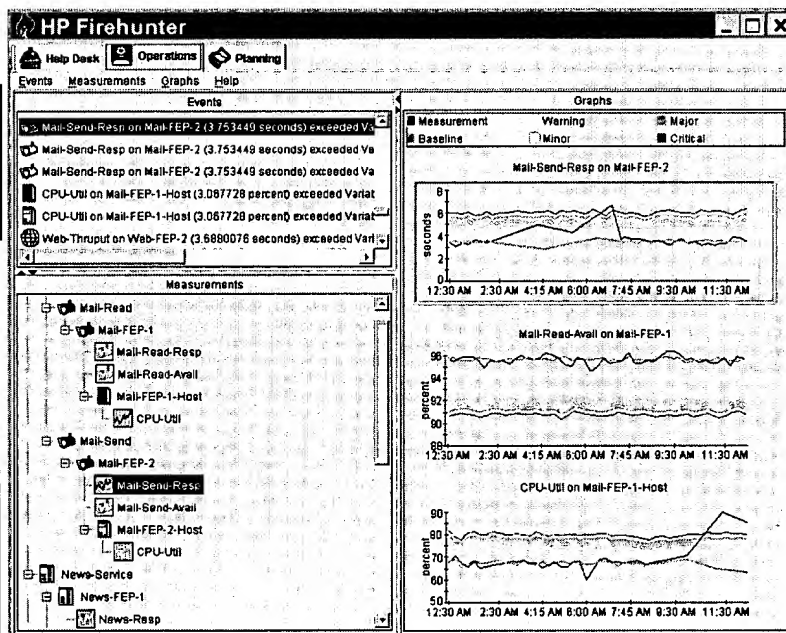
- Help Desk—The help desk window consists of icons and annotation views for each modeled service you provide, such as VPN services, voice-over-IP, or web-hosting services. When a problem arises in a service, you can create an annotation in the Operations window



that briefly describes the problem and when it might be fixed. Firehunter displays that information in the Help Desk window, where it appears in the corresponding service annotations box. The help desk user then refers to that information when answering calls about problems customers are experiencing with certain aspect of your services, such as determining when a service will be restored. For quick

reference, the service's icon displays a color that represents the current health of that service.

- Operations—The Operations window consists of the following views:
  - Events—displays notices about measurement values that have crossed threshold boundaries which you have configured.
  - Services—displays the service model, which includes all services being managed by Firehunter, along with their components and the measurements being taken for each.
  - Graphs—displays information about measurements in time-based graphical form.



Events are generated when measurements exceed service oriented thresholds. Actions can be automatically initiated

Service-based views of the ISP environment is built upon component performance measurements including response times, utilization, and availability

Service models and measurement tests can be customized to accurately reflect customer specific environments.

Real-time web access to events and graphs

Measurement graphs illustrate current and/or historical information (baselines).

Highly customizable, graphs can be saved as reports, and data can be exported into other tools

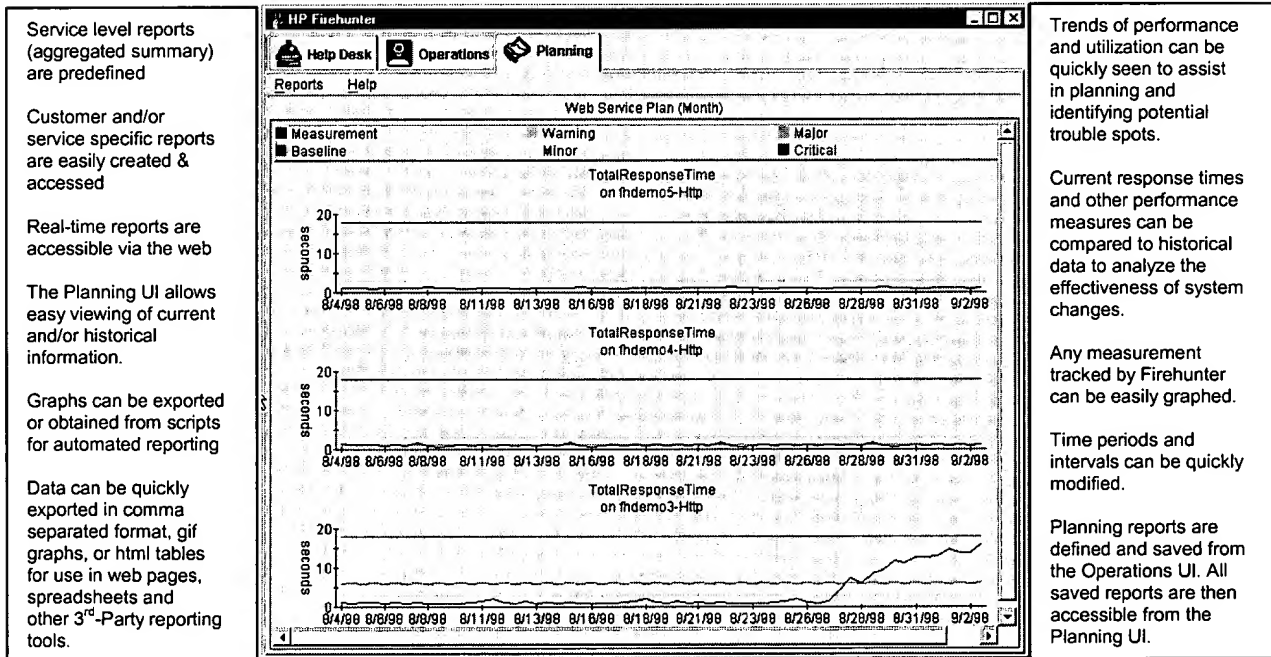
Thresholds can be user defined or automatically created to reflect historical baselines and trends.

Graphs, Events, Measurement, and Services are all contextually linked.

When an event occurs, Firehunter responds by writing information about it to the Events view. You can see exactly where the event occurred by expanding the service model in the Services view. You can then view a graph of the measurement that generated the threshold event in the Graphs view. You can also set up Firehunter to display services and graph information automatically when an event occurs.

In addition to viewing events, services, and measurements, you can create custom reports from the Operations window. Although Firehunter ships with a set of pre-defined reports, you can create your own to provide information about the system that the pre-defined reports do not contain or do not present in the format you want. Whether you use pre-defined or custom reports, using them enables you to communicate information about trends in services to business managers and planners.

- Planning—The planning window consists of the Reports view, in which Firehunter displays the report you select. (Selectable reports are defined in the Operations Window.) You can export



any of these reports or the data they contain into either Comma-Separate-Value (CSV) files so the data can be incorporated into customized reports, or into HTML files so that they can be viewed through any web browser.

- **Admin Console**—In addition to the Help Desk, Operations, and Planning Views in the Client GUI, there is an Administrative Console which can be used to configure many aspects of Firehunter. The admin console enables you to customize Firehunter for your service assurance objectives. You first use it to run the discovery process, set system parameters, and verify your service model. Then, you may use it to further modify the service model by defining thresholds, and adding measurements and tests. You can easily include data from other management tools or their data stores to leverage into Firehunter management information you already collecting through other means.
- **SLAs** -- Firehunter also gives you the ability to represent, monitor and report on Service Level Agreements (SLAs). Though not a part of the GUI to-date, you can configure Service Level Agreements to monitor and manage to the guarantees you have established (or would like to establish) with your customers. Firehunter is highly flexible in the conditions that can be applied to any measurement in formulating SLAs, and it can be used to identify new, viable SLAs at differentiated service levels that you can offer your customers. Reports can be scheduled, are content-customizable and easily generated, and can be viewed with a web browser.

In addition to Firehunter's GUI, you can use other applications to view much of the data gathered. In addition to the graphs and data already discussed, you can use your browser or a CSV-compatible application to view Firehunter data and SLA results.

## **DMS**

The DMS (Diagnostic Measurement Server) resides at the mid-tier and performs analysis and correlation of agent-collected data. It provides



the detailed information about the quality of your delivered services that is viewable through the Client GUI. The DMS performs the following functions:

- discovers the location of the ISP services to be managed based on the service model and its discovery engines. The service discovery module writes the information it gathers into the service model, which is a configuration file. The Service Model is a representation of your managed services and their components, and is used to control Firehunter's behavior.
- stores and maintains all measurement data Firehunter gathers through its agents, and calculates thresholds and baselines for these measurements. Baselines characterize "normal" levels of service quality for given hours of the day for each day of the week. The DMS also deploys the agents that take these measurements, reducing the administration associated with large numbers of remote agents.
- determines the health of each service model component. To do this, the DMS compares measurements to static or variable (baseline) thresholds which you can configure for your service components. Threshold crossings are used to evaluate the health of these components. By default, the health of the component reflects its most severe event and is reflected up the service model tree through status propagation.
- implements defined actions when threshold events occur, indicating anomalies that require attention.
- reports on your ability to meet service level agreements you have established with your customers and helps you identify new SLA opportunities you can offer your customers.

## **Agents**

An agent executes tests to gather measurements of your services' performance, availability, and other quality levels, as defined in the service model. Each agent consists of the following main components:

- Tests determine which measurements the agent should take.

- The agent controller determines which tests the agent runs, how frequently it runs them, and where it sends the measurements.

Agents take the following two types of measurement samples:

- *Active* measurements create a stimulus to measure the service's response. For example, an agent might emulate a client and interact with its server.
- *Passive* measurements monitor log files and system utilities to gather additional information available only at a server. Firehunter gathers passive measurements using an agent that resides on the same machine as some of its service components.

As an agent collects measurement samples, it sends them to the DMS at intervals defined in the service model. The DMS includes an agent, but you can also install and run agents on remote hosts, as described in the following section. This is required to take passive measurements on hosts without a DMS as well as for gathering link-to-link network and other service characterizations.

---

## Scalability

Firehunter offers you the following scalability options that provide the versatility and expandability you need for your specific infrastructure and service assurance goals.

### **Adding remote agents**

The Firehunter DMS includes its own agent, but can gather data from multiple, remote agents as well. You may desire to install additional remote agents for one or more of the following reasons:

- To better simulate the end-to-end service usage experience, by taking measurements from points along the user's network traffic path.

- To take passive measurements on services running on other servers. To perform a passive measurement, the agent must reside on that server.
- To perform active measurements from alternate points in the environment. A comparison across several perspectives is helpful in troubleshooting and isolating the source of a problem. For example, a firewall or proxy may be excluded in one measurement path, but included in another.
- To do workload balancing by offloading some measurements from a DMS's local agent to one or more remote agents.

### **Adding DMSs with Firehunter/PRO**

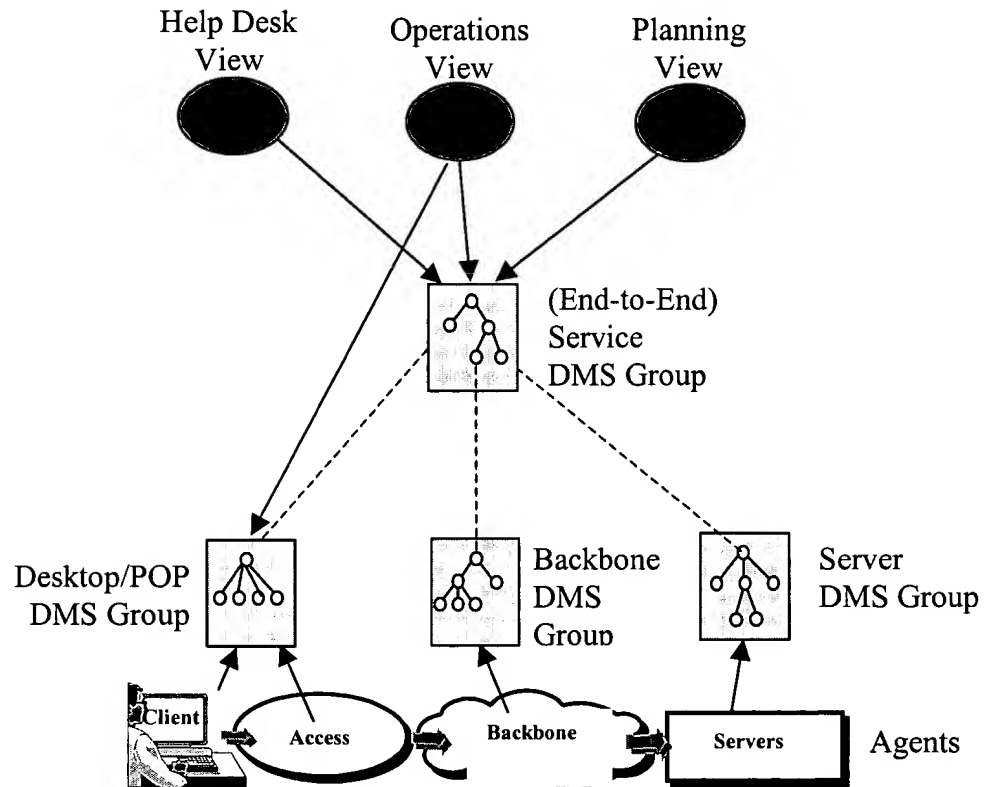
As your customer base and service offering grows, you may want to expand your deployment to include multiple DMSs for any of the following reasons:

- Geography. You want measurements to be collected in close proximity to the resources being monitored.
- Service mix. You may want to take different measurements on different machines due to the types of services they provide. For example, you may want all your email servers controlled by one DMS, and all your web servers by another.
- Volume. You may simply have so many servers and services that your DMS becomes congested.
- Organizational. Your services may each be administered by different teams who want to control their own service models for the portion of the environment for which they are responsible.

When you add DMSs to your system, you add them hierarchically; one DMS is the “parent” that receives summary data from the other DMSs. However, each DMS may run autonomously. Architecturally, there is no limit to the number of DMS levels you can establish. Also, remote agents may tie in at any level in the DMS hierarchy, and each agent may forward data to several DMS's, as specified by the service model configured for each DMS and the measurements required for

each. An example of multiple distributed DMS's is shown in the diagram below.

Once a DMS hierarchy is established, the Client GUI and reports can navigate the multiple tiers of DMS parents and children transparently. Through the connections established, each DMS contributes its own portion of the overall service model.



---

## Summary

This document has described the three tiers of the Firehunter architecture—the agents, the DMS, and the GUI clients. The agents take measurement samples of specified ISP services. The DMS is the collection point for measurement data. It stores historical data, establishes baselines, generates threshold events, and determines system health. The client GUI displays the measurement, threshold, and baseline information thus enabling you to monitor the health of your ISP system from one window. The Firehunter/PRO implementation adds additional capabilities for distribution and scalability to manage large distributed Internet environments. It also provides support for extension of the core Firehunter product including the ability to add new service models and integrate with other management products (network element management, systems management, etc.) to compliment Firehunter's Internet service management capabilities.

## Service Model and Service Health

Firehunter uses a unique service model to integrate Internet components into a tree-like structure that allows the entire service environment to be easily visualized and managed. The service model simplifies management by aggregating data into meaningful information and enabling faster fault detection, isolation and resolution. This chapter describes the service model and service health and explains how each works.

## Anatomy of the service model

Each ISP service (email, web, news, and so forth) is represented in the Firehunter Service Model as a tree like structure. At the root is the service itself, followed by the servers which implement the service. Each of these servers in turn is composed of the supporting service elements (application software such as Post Office Protocol/POP3, Simple Mail Transfer Protocol/ SMTP, operating system, and network interfaces), that support delivery of the service.

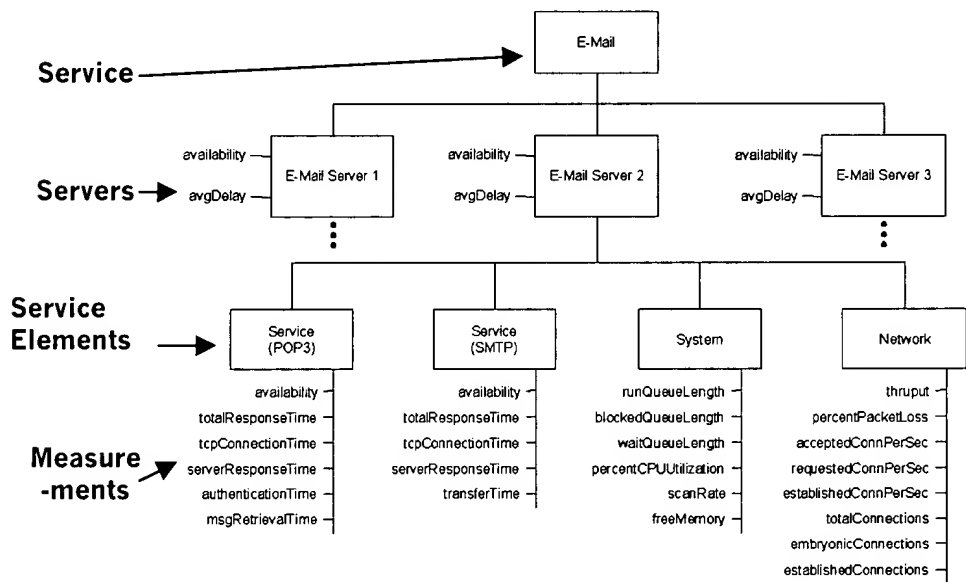


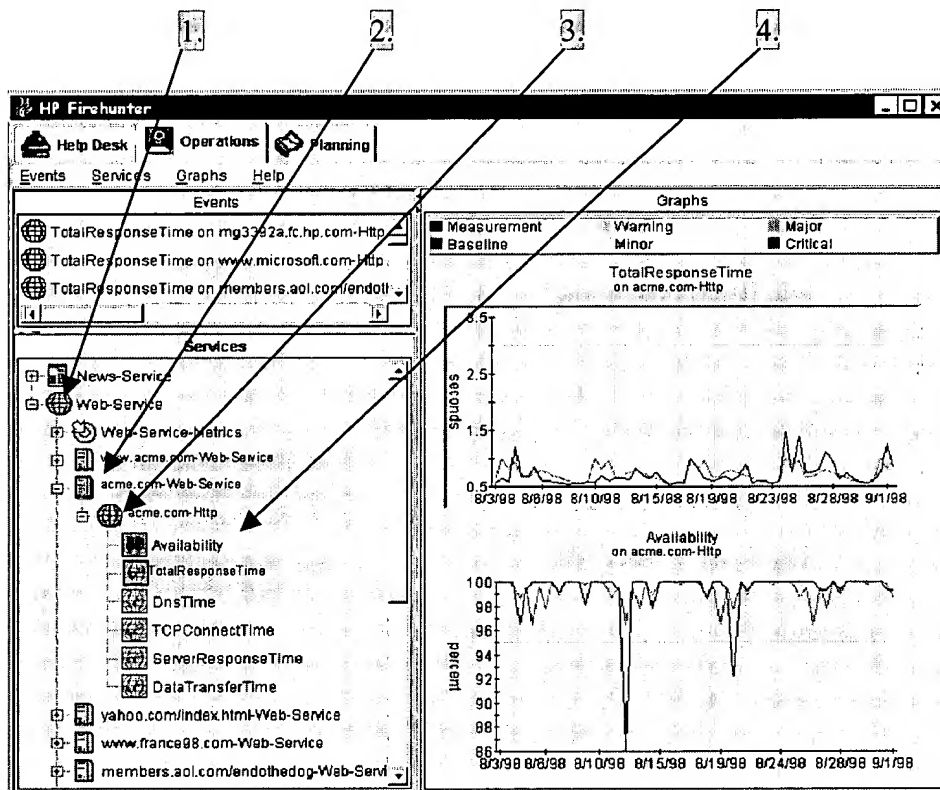
Figure 1

Finally, note that dependent elements of the infrastructure (domain name service, authentication servers, and backbone throughput, for example) are also service elements tied into the service model. At the leaf nodes of this service tree are the actual measurements that monitor fundamental aspects of the health and performance of each of the components represented higher up in the tree. Figure 1 shows a service model for hypothetical email service.

A typical Firehunter service model display is shown in the lower left pane of Figure 2. Performance measurements, for selected components, are shown in real-time graphs in the right-side pane of the display. The display provides an instantaneous view of overall health using color-coded icons to indicate the status of services and components. It allows the user to quickly navigate to the source of a problem – even in the most complex environments. The standard Firehunter product includes service models for managing E-mail, Web, News, DNS, and Network services. In addition to these standard features, the product is fully customizable, allowing easy addition of measurements and service models to enable management of new service offerings.



The service model tree is represented in the Firehunter GUI as a nested file structure. With each service (e-mail, news, web, etc.) at the top **1**, service elements including instances of the service **2**, and supporting protocols such as HTTP **3** tiered below. At any level in this service tree are the actual measurements that monitor fundamental aspects of the health and performance of each of the components represented **4**. Actual measurements of service element performance, such as “Availability” of the HTTP (web) service instance on server “acme.com” are shown in the graphs in the right hand pane of the GUI display.



**Figure 2**  
Service Model Example

## Building the service model

Your service model is automatically “discovered” during the initial installation process based on a template that defines the types of service elements and measurements being modeled. During that time, Firehunter uses several discovery engines to locate the following parts of your system:

- IP Hosts. You can limit the scope of discovery by entering an IP address list. Firehunter will only discover IP hosts within that address range.
- Web services
- News services
- SMTP (Simple Mail Transfer Protocol) services
- POP3 (Post Office Protocol) services
- Email pairing between the discovered SMTP and POP3 services
- DNS (Domain Name Service) services

The discovery engines may be periodically re-run after your installation so that changes in your environment may be incorporated in the service model. You may also need to manually edit the service model; for example, you may want to add a node to the list of discovered nodes, change measurement points, or to add a new test.

The Firehunter/PRO product is fully extensible, with the ability to add and modify service models, views and reports for tailored ISP service management solutions.

- HP Consultants can add new service models (Web Hosting, VPN, VOIP, etc.), discovery engines, and provide custom views and reports.
- Users can add measurements, thresholds, baselines, and status propagation rules. They can also configure thresholds, test frequency, measurement views and reports.

---

## Service health

The concept of service health is based performance baselines and thresholds. By sampling data over a period of time, Firehunter builds up a baseline, a historical record of how a measurement behaves (its “normal” performance). Once the baseline has been determined, an envelope around the baseline can be set. The outer edges of this envelope represent high and low thresholds. Thresholds can be variable (following baseline variances) or static (a fixed level above or below the baseline). Events are generated only on threshold crossings to alert operators of service anomalies. Actions (pager, e-mail notifications or scripts), triggered by events, may be used to link the system with supporting tools such as Trouble Ticketing or system and element management systems. Using Firehunter, ISPs can quickly assess service health by looking at the color of service icons in the service model. Thresholds and Actions enable ISPs to manage by exception. They can set thresholds to limit events and actions to only those service anomalies that require immediate attention.

Thresholds mark limits within which measurements should fall to ensure quality of service. For each measurement there may be multiple threshold boundaries—warning, minor, major, and critical. Each threshold boundary represents problems in the service and indicates how critical those problems are. For more information, see the next chapter.


When a measurement crosses a threshold boundary, Firehunter generates an event that is displayed in the Events view of the Operations window and invokes any defined actions. The event includes the following information:

- measurement that exceeded a threshold boundary
- where in your system the measurement was taken
- exact value of the measurement
- severity level of the crossed threshold boundary
- threshold type
- threshold boundary value

- date and time the event occurred

Events will vary in their severity based on which threshold boundary is crossed. In addition, each event has an associated icon that appears in a color representing the severity of the event:


- Red—critical
- Orange—major
- Yellow—minor
- Cyan—warning
- Green—normal
- Light blue—unknown


The severity and number of events influences the health of the components and services to which the events are related. Firehunter propagates the most severe status to the top of the service's hierarchy  in order to quickly alert you of any problems with service health (see Figure 3).

---

## Using the service model

The service model greatly facilitates fault diagnosis. When a problem is detected, the service model provides structure to the drill-down process used to isolate the root cause of the problem. For example (see Figure 3):

Suppose Web Service response time is below normal. As an operator, you are alerted to this problem by Red Web Events (DNS Time and Total Response Time) .

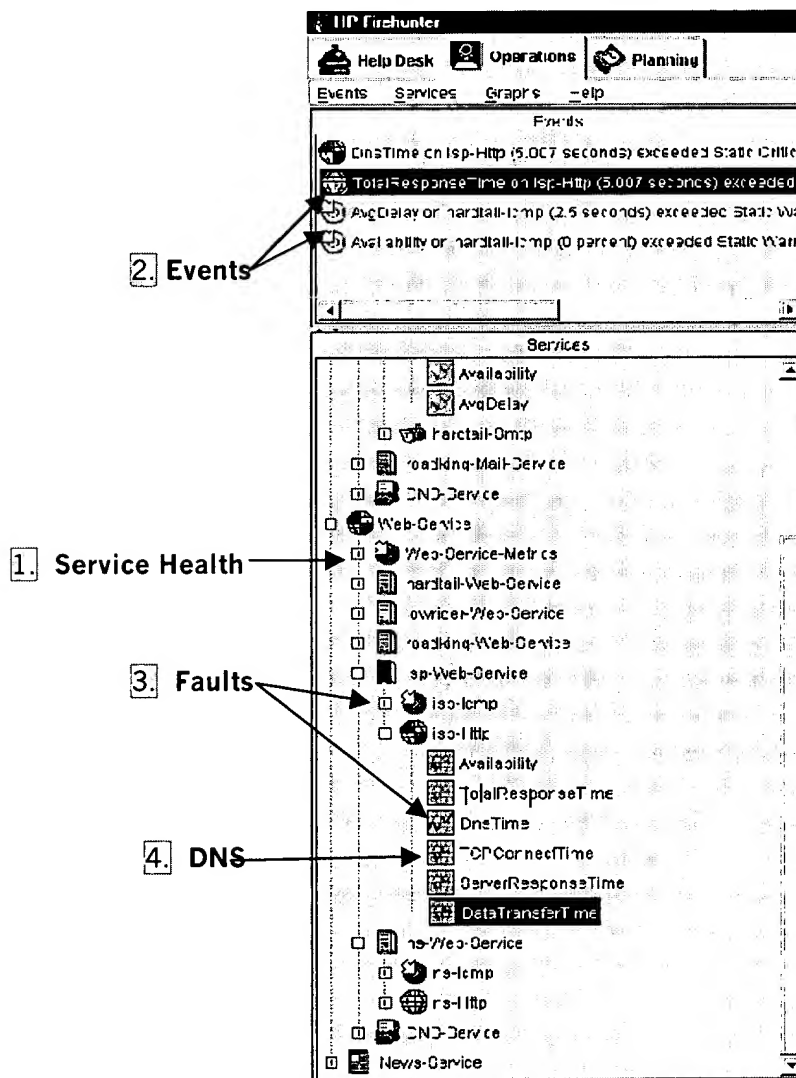
Clicking on the events expands the service model to shows which measurements were detected to be at fault (Web Service, server isp-Web Service, service process isp-Http, and measurements on Total Response Time and DNS Time) .

As you explore the service model, you see that only one web server (isp-Web Service) is experiencing the problem and that the problem is with HTTP Total Response Time. This immediately eliminates many aspects of the

infrastructure that might have contributed to the performance problem, such as backbone performance, or authentication server problems.

Further examination shows you that the DNS (naming) server is experiencing greater than acceptable response times [4]. This increase in response time appears to be the cause of the problem with Web Service Total Response Time.

By examining the individual components of the Web service you can easily see that the related components including Server Response Time, Data Transfer Time, and TCP Connect Time on server isp-Web Service, are performing properly. Thus further isolating the problem to a DNS name resolution delay.



**Figure 3.**

By selecting the measurements that show fault conditions and generating measurement graphs we can quickly see the correlation <sup>1</sup> between the Total Response Time problem and the DNS Time and where thresholds <sup>2</sup> were crossed (see Figure 4.).

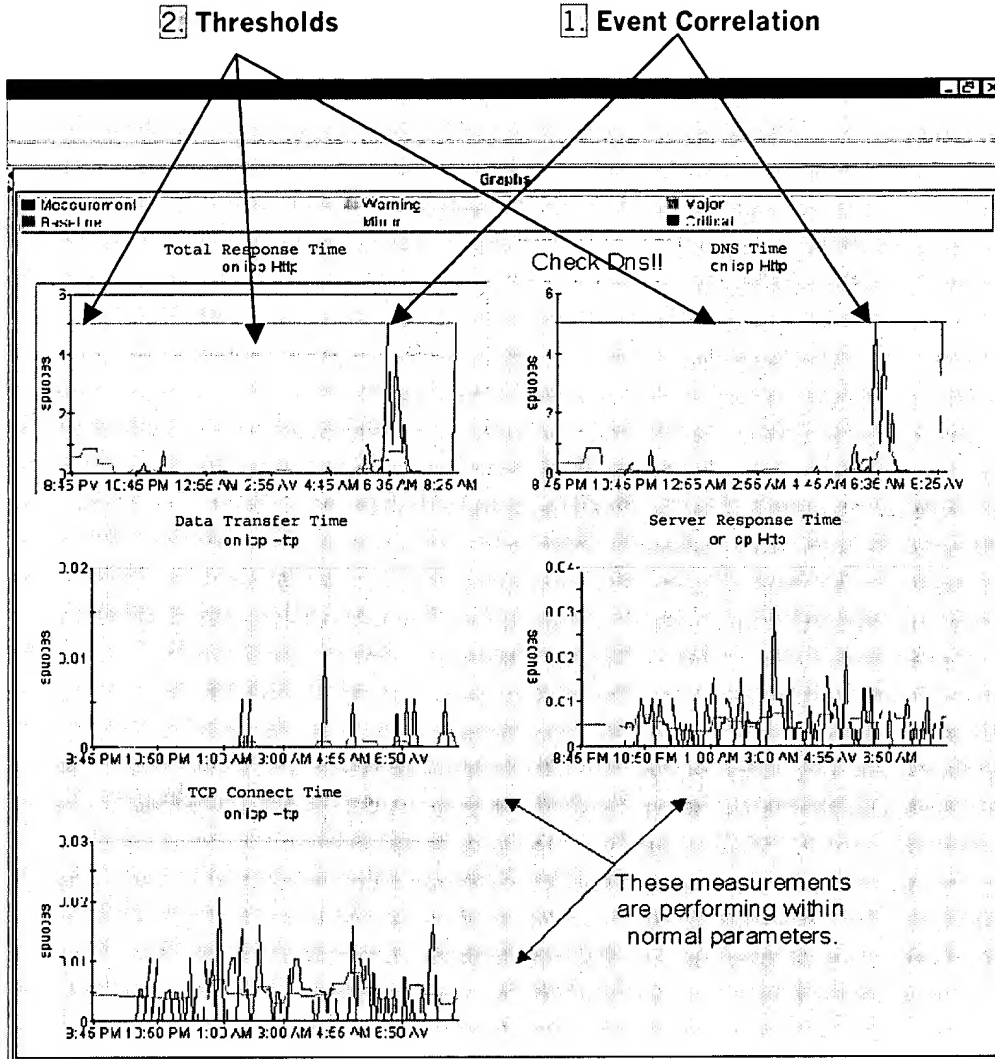


Figure 4.

## Summary

Firehunter uses unique discovery and modeling technologies to enable easy visibility of the Internet service environment and efficient fault detection and diagnosis. Service Models represent the structure of a service and the components upon which it depends. This structure helps identify critical performance measurements and thresholds. In addition, Service Models facilitate diagnosis by providing structure to the diagnostic process. The graphical service model view provides a real-time, top-down picture of service health. With Firehunter's service-based approach, ISP's can effectively manage quality of service to improve business results.



## Tests and Measurements

A Service Model is only as good as the data it correlates. Measurement data makes up the fundamental building blocks of the entire system. To provide meaningful events and trending data, measurements have to be accurate, timely, and focused at the correct targets. This chapter provides the information you need to understand the Firehunter measurements and how they can be interpreted to locate problems with your system.

---

## Measurement types

As explained in Chapter 1, Firehunter takes two different types of measurements:

- Active measurements explicitly stimulate traffic to networks, servers, and service applications to assess various metrics of interest about these elements. These measurements are used primarily to assess service quality.
- Passive measurements use instrumentation built into network, server, and service application components to track real subscriber traffic. Besides being useful for service quality assessment, passive measurements can also provide information about resource and service usage, which is crucial for detailed problem diagnosis and capacity planning.

Combining active and passive measurements provides information that is critical for effective operational monitoring and capacity planning for ISPs. To maximize the effectiveness of the measurements and to minimize their impact on the environment, Firehunter's flexible service model allows you to easily configure where agents are placed in the system and how often the measurements are taken. For example, you may want to take passive measurements rather than active measurements to avoid impacting quality of service through additional traffic, and to gather more useful data for problem diagnosis. To do so, you will need to add remote agents to your system.

---

## Basic tests and measurements

Firehunter provides basic service quality, network, infrastructure service, and server measurements and tests as described in the following sections. You can also incorporate other measurements and tests that you are already taking through other products into Firehunter. See “Adding measurements and tests” later in this chapter.

### Web service tests and measurements

Web service measurements assess the web service’s availability and performance. Firehunter includes the following basic tests, which you can use and/or add to:

#### HttpPortStats

This passive test measures statistics on the HTTP port (default 80). It includes the following measurements:

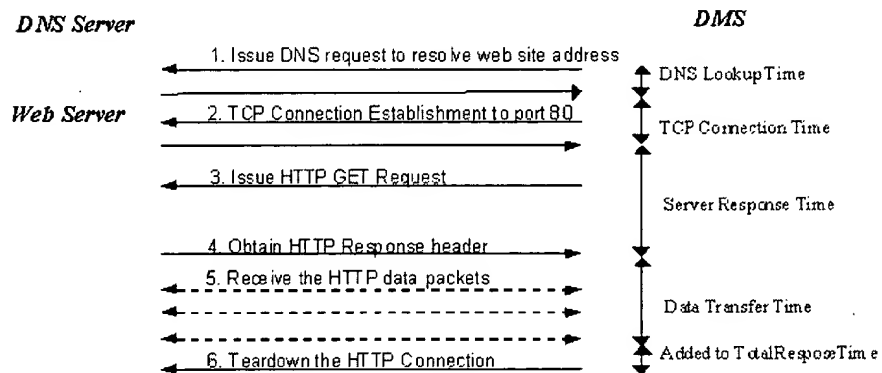
Measurement	Description
TotalConnections	Total number of connections in the service’s port
EmbryonicConnections	Number of connections started, but not complete
EstablishedConnections	Number of active connection with the server

#### HTTP

This active test measures web availability and web page retrieval time. It provides response time breakdown into subcomponents. It includes the following measurements collected via the timing shown in the figure below:

Measurement	Description
Availability	Determines if the URL is available
TotalResponseTime	Total time of DnsTime, TCPConnectTime, and ServerResponseTime
DNSTime	Time of DNS name lookup
TCPConnectTime	Time to establish TCP connection with the service
ServerResponseTime	Time from TCPConnectTime until the server responds
DataTransferTime	Time from when the web page is requested until the web page is completely returned. Only the base HTML document is transferred; images, references, or links are not loaded or followed.

## Web Test Component Measurements



**Figure 1 - HTTP Test Components**

### Email service tests and measurements

Email service measurements assess the ability of your subscribers to send mail to other subscribers and to receive mail from the email servers. Firehunter includes two passive tests, two separate active tests to assess mail delivery and retrieval, and a round trip test as described in the following sections:

#### Pop3PortStats

This passive test measures statistics on the POP3 port (default 110). It includes the following measurements:

Measurement	Description
-------------	-------------

TotalConnections	Total number of connections in the service's port
EmbryonicConnections	Number of connections started, but not complete
EstablishedConnections	Number of active connection with the server

### **SmtpPortStats**

This passive test measures statistics on the SMTP port (default 25). It includes the following measurements:

Measurement	Description
TotalConnections	Total number of connections in the service's port
EmbryonicConnections	Number of connections started, but not complete
EstablishedConnections	Number of active connection with the server

### **SMTP**

This active test measures the availability of the mail server and performance of sending mail. To run this test, you must have one dummy email account on the target server. Since this test sends messages to this mailbox, you must provide a mechanism to automatically delete the messages.

It includes the following measurements collected via the timing shown in the figure below:

Measurement	Description
Availability	Determines if the SMTP daemon is

	available
TotalResponseTime	Total time of connection and sending of messages
TCPConnectTime	Time to establish TCP connection with the service
ServerResponseTime	Time from TCPConnectTime until the server responds
TransmissionTime	Time for message to reach destination

## Smtplib Test: Component Measurements

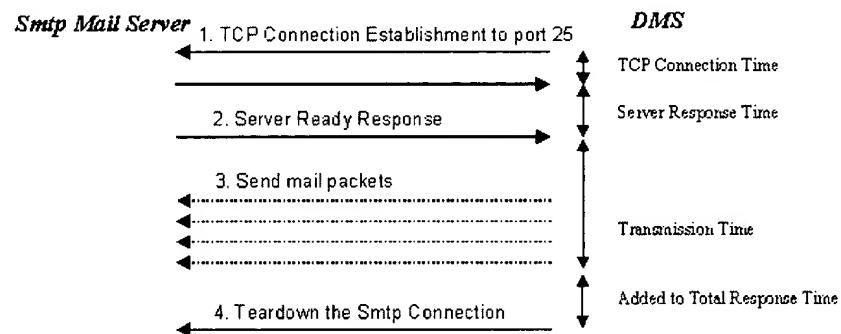


Figure 2 - SMTP Test Components

### POP3

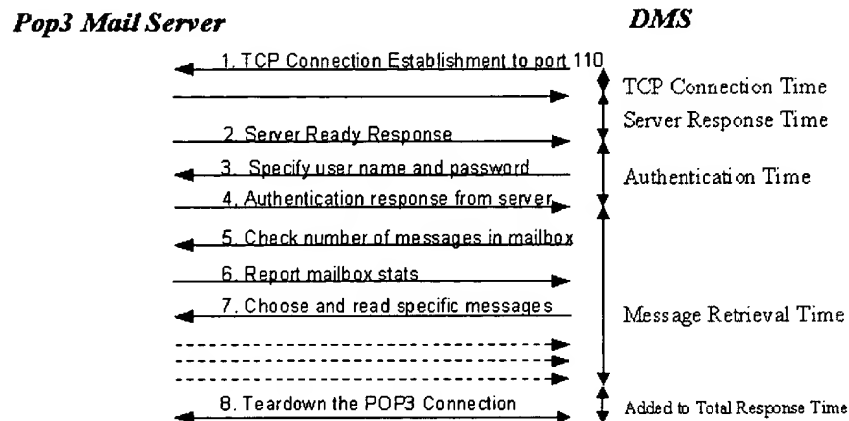
This active test measures availability of the mail server and performance of retrieving mail. To run this test, you must have one dummy email account on the target server. The test will seed the mailbox with a message and measure its retrieval.

It includes the following measurements collected via the timing shown in the figure below:

Measurement	Description
Availability	Determines if the POP3 daemon is available
TotalResponseTime	Total time of connection and retrieval of message
TCPConnectTime	Time to establish TCP connection with the service
ServerResponseTime	Time from TCPConnectTime until the server responds
AuthenticationTime	Time to authenticate user during login
MessageRetrievalTime	Time to retrieve message from mailbox



## Pop3 Test: Component Measurements



**Figure 3 - Pop3 Test Components**

### EmailRoundTrip

This active test sends a message to an SMTP server, which forwards it to a POP3 server. The test then retrieves the message from the POP3 server. Measurements are taken of the SMTP and POP3 connections and on the total round trip time as described in the following table:

Measurement	Description
Availability	Determines if the SMTP daemon is available
TotalResponseTime	Total time of connection and sending of messages

TCPConnectTime	Time to establish TCP connection with the SMTP service
ServerResponseTime	Time from TCPConnectTime until the SMTP server responds
TransmissionTime	Time for message to reach destination
Availability	Determines if the POP3 daemon is available
TotalResponseTime	Total time of connection and retrieval of message
TCPConnectTime	Time to establish TCP connection with the POP3 service
ServerResponseTime	Time from TCPConnectTime until the POP3 server responds
AuthenticationTime	Time to authenticate user during login
MessageRetrievalTime	Time to retrieve message from mailbox
AvgEmailRoundTripTime	Average round trip time for email messages
MaxEmailRoundTripTime	Maximum round trip time for email messages
TimedOutEmailMsgs	Number of email messages not received before timeout period

### **News service tests and measurements**

News service measurements are quite similar to web service measurements described earlier, as described in the following sections:

### NntpPortStats

This passive test measures statistics on the NNTP port (default 119). It includes the following measurements:

Measurement	Description
TotalConnections	Total number of connections in the service's port
EmbryonicConnections	Number of connections started, but not complete
EstablishedConnections	Number of active connection with the server

### NNTP

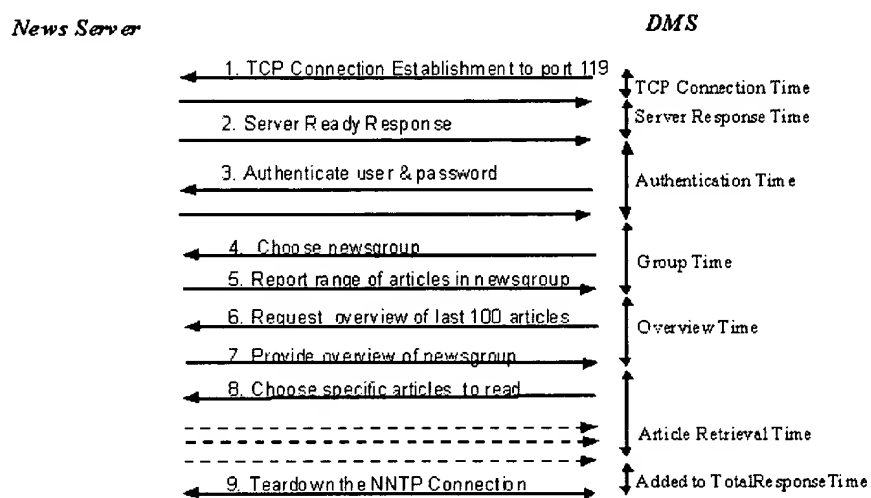
This active test measures the availability of the news server and performance of reading news.

It includes the following measurements collected via the timing shown in the figure below:

Measurement	Description
Availability	Determines if the NNTP daemon is available
TotalResponseTime	Total time of connection and reading
TCPConnectTime	Time to establish TCP connection with the service
ServerResponseTime	Time from TCPConnectTime until the server responds
AuthenticationTime	Time to authenticate a use during login
GroupTime	Time taken by the news server to set the current newsgroup to the newsgroup being

	used for the NNTP test
OverviewTime	Time to retrieve an overview of all news articles in the news group
RetrievalTime	Time to retrieve a news article in the news group

## News Test Component Measurements



**Figure 4 - NNTP Test Components**

## Network tests and measurements

To obtain an end-to-end perspective of service quality, web, email, and new service measurements should be complemented with network measurements. For services that use TCP for reliable communication, a useful network performance metric is throughput, defined as the rate of reliable transmission of packets between a source and a destination. The throughput achievable between any source-destination pair is a complex function of several factors such as the characteristics of the source and destinations TCP implementation, processing capabilities of the source and destination, bursts of packet loss, round-trip packet transmission delays, and so forth.

Network tests are described in the following sections:

### TCPConnectionRate

This passive test measures TCP connections to and from a specific host. The measurement is the aggregate for all ports on the servers, indicating the number of connections that have been accepted, established, and closed. This test includes the following measurements:

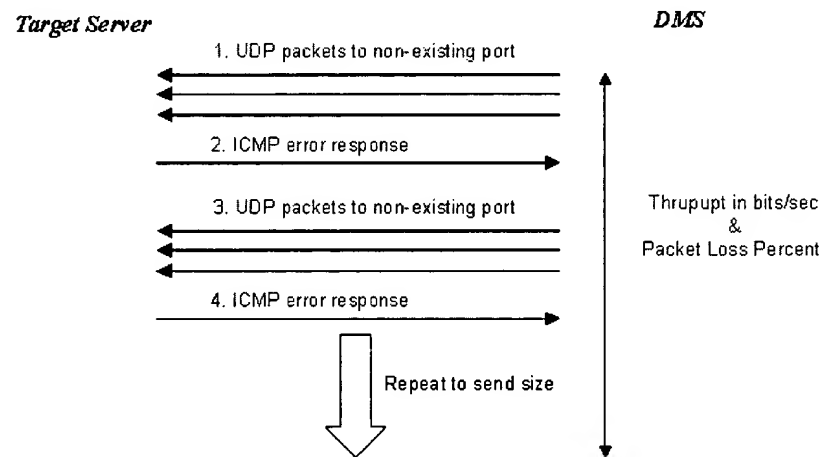
Measurement	Description
AcceptedConnectionsPerSec	Number of connections accepted per second
RequestedConnectionsPerSec	Number of outbound connections requested per second
EstablishedConnectionsCounter	Number of connections in the ESTABLISHED state

**Thruput**

This active test simulates TCP throughput and re-transmissions. It includes the following measurements collected via the timing shown in the figure below:

Measurement	Description
Thruput	Throughput in Kbits per second
PktLossPercent	Percentage of TCP packtes that required retransmission

**Thruput Test:  
Component Measurements**



**Figure 5 - TCP Thruput Test Components**

### **Icmp**

This active test measures network connectivity and delay. It includes the following measurements:

Measurement	Description
Availability	Determines if the host is reachable with an ping()
AvgDelay	Average time for a request (response time)

### **DNS measurements**

To assess DNS performance as perceived by your subscribers, you must take measurements that track address resolutions requested by subscriber applications. Firehunter offers one active test described below:

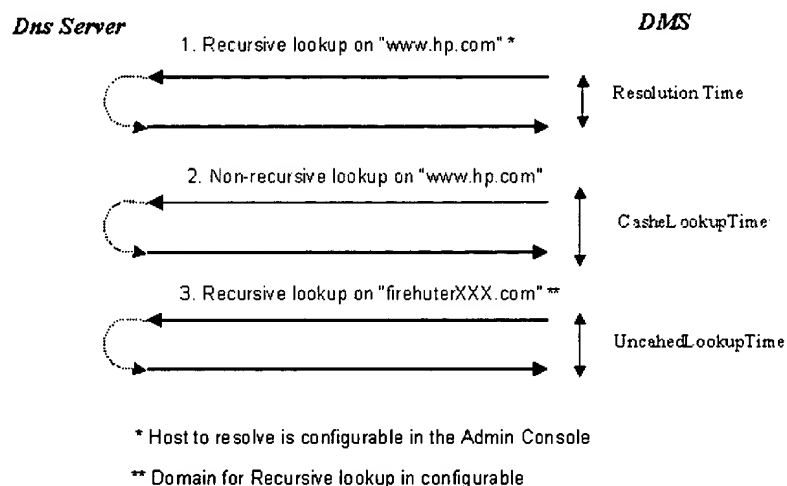
#### **DNS**

This active test measures the availability of the DNS daemon, ability to resolve a host name, and response times for cached and uncached name. It includes the following measurements collected via the timing shown in the figure below:

Measurement	Description
Availability	Determines if the DNS daemon is available
UnCachedLookupTime	Response time for recursive lookup of a non-existent hostname
CachedLookupTime	Response time for looking up a name in the DNS cache
Resolved	Ability to resolve the address of a hostname

ResolutionTime	Response time of the name resolution
----------------	--------------------------------------

## Dns Test: Component Measurements



**Figure 6 - DNS Test Components**

### Server measurements

Measurements taken on ISP servers can yield a wealth of information that is critically important for isolating problems. Firehunter's agents executing on ISP servers passively track various statistics of interest in the following test:



### Vmstat

This passive test reports statistics about process, virtual memory, and CPU activity. It includes the following measurements:

Measurement	Description
RunQLength	Number of processes waiting to run
BlockedQLength	Number of processes blocked, waiting for resources
FreeMemory	Amount of free memory pages at the time of the test
PercentCPUIdle	Time percentage the CPU is idle

---

## Adding measurements and tests

You may be collecting data with your own custom measurements or subscribing to a service that collects data for you. You can add these measurements to your Firehunter Service Model to incorporate in your graphs and reports.

Measurements are integrated into Firehunter as a test. A test is an executable or script that is run (passing it command line parameters) by the Firehunter agent service. There may be a different test executable/script for each supported agent platform.

Once a test has been properly implemented, configured and installed, Firehunter will execute the test at the scheduled time (as defined by the Frequency attribute for the test in the Service Model), transport the test's measurements to the DMS, and load the measurements into the DMS. Once the data is loaded into the DMS it is treated like all other Firehunter measurements. Baselines will be calculated, thresholds can be generated, service health is affected by the

measurements, actions taken on the thresholds and service health, reports defined for your data, etc.

For example imagine that a large corporate customer want to add a "mean time to repair" metric to their Service Level Agreement. If this information is available programmatically from your trouble ticketing system a test may be created to collect this data. Once collected it may be included in a SLA template for that customer as any other Firehunter measurement.

## Thresholds and Baselines

With Firehunter you won't be overwhelmed with volumes of non-essential management events. It's distributed service management system works continuously to gather and correlate new measurements across multiple domains. New measurements are compared to established baselines, that represent expected "normal" behavior, and to associated user defined thresholds. Events are generated only on threshold crossings to alert operators of service anomalies. Actions may be added, in response to events, to trigger special notifications (pager, e-mail, etc.) or scripts that can initiate corrective system configuration changes. Actions may also be used to link the system with supporting tools such as Trouble Ticketing or system and element management systems. Using Firehunter, ISPs can manage by exception, quickly analyzing and intelligently responding to any deviations from "normal" behaviors, and efficiently addressing growth rates and other trends.

Thresholds enable you to define warning boundaries relative to normal service behavior so that Firehunter can provide early detection of deviations and alert you through use of events and/or actions. Although Firehunter comes with predefined thresholds, you will most likely want to modify these thresholds for your specific system. Methods used for baseline calculation are also configurable, however it is less likely that you will want to modify the default baseline parameters.

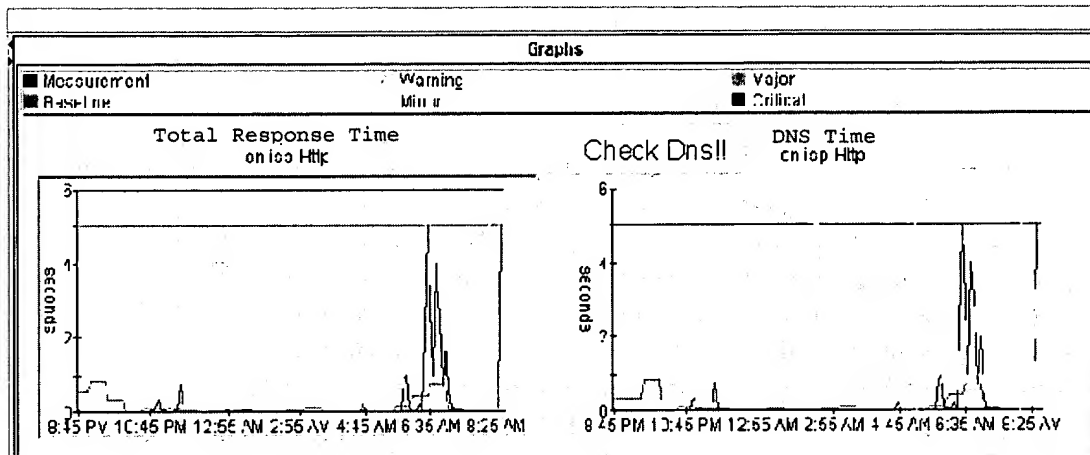
For each measurement made in your system, you may choose to set up to four threshold boundaries, one for each level of severity shown in the GUI (warning, minor, major, and critical). Each threshold may also be either static or variable. This chapter provides the information you need to decide how to best set up baselines and thresholds in your environment.

---

## Setting static thresholds

In a static threshold, the defined boundaries are fixed points. For example, you may want to monitor your disk space and be alerted when it reaches a certain level. Static thresholds are used to simplify measurement graphs when a fixed performance level or percentage is

sufficient for monitoring service anomalies or trends – for example an Availability threshold might be set at a static 95% to alert operators of an unacceptable trend in server availability (see Figure 1).



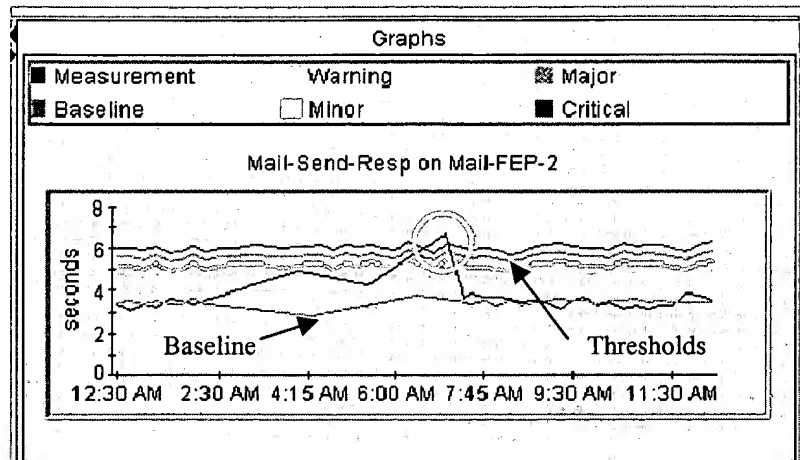
**Figure 1.** – Example of Static Threshold Use

## Setting variable thresholds

Many metrics that determine service health vary considerably over time. These variations don't necessarily indicate abnormal conditions; often they are due to normal periodic changes in the use of the service. These variations are usually cyclic and occur on a daily, hourly, or weekly basis. For example, the range of acceptable values for the POP utilization measurement varies depending on the time of day and the day of the week—it will have different values at 2:00 am and 7:00 pm on the same day or at the same time on Monday and Sunday. If you try to use static thresholds, either problems will be left undetected or false alarms will become rampant.

To develop thresholds that can handle the normal cycles of measurements prevalent in the ISP environment, Firehunter incorporates variable thresholding. With a variable threshold, you define your threshold as a deviation from your normal activity rather than as a static value. To determine your normal activity, Firehunter

samples data over a period of time and builds a baseline, a historical record of how a measurement behaves. Once the baseline has been determined, you set an envelope



**Figure 2. - Measurement Baselines and Thresholds**

around the baseline representing high and low thresholds. When actual measurement samples cross either of these thresholds, an event is generated. You can set up multiple events using multiples of the standard deviation, percentiles, and so forth (see Figure 2 and Figure 4).

To build a baseline for a measurement, Firehunter must have been running on your system long enough to build a history for that measurement. Firehunter needs this information to calculate the baseline, but you also need the information to understand what parameters will best represent your system. HP recommends collecting at least four weeks of data before building baselines and setting up variable thresholds.

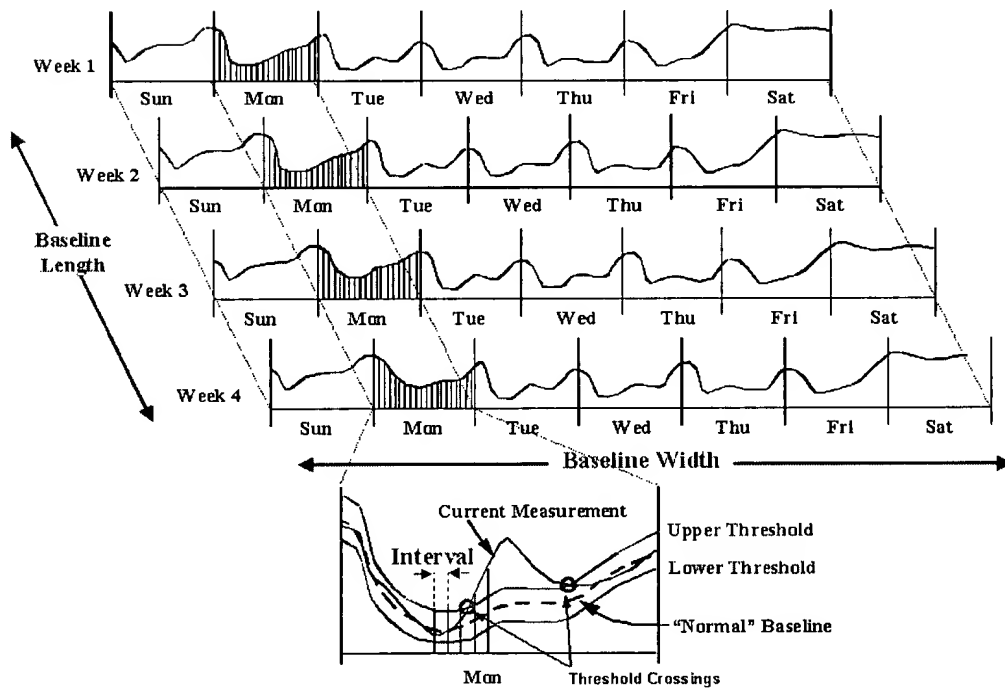
When you do build a baseline, you need to provide the following parameters to Firehunter (see Figure 3).

- Baseline Width (period)—the length of the cycle between similar measurements. Based on experience, we have set the baseline

width at one week. This best represents the normal business cycle for Internet usage. From week to week, your system will normally experience similar demands at the same time on the same day of the week. In some cases, your period may be daily; that is, every day has the same demands at the same time. Firehunter fixes the baseline width at 1 week.

- **Baseline Length (history)**—the length of time over which measurement samples will be used to determine the baseline. To a large degree, the length of the history determines how accurate the baseline is and in turn affects the degree of confidence you can have in your thresholds. Configurable parameter in Service Model (default is four weeks).
- **Interval (window)**—the length of time over which data is analyzed to calculate baseline statistics. This window is usually expressed in minutes or hours and depends on how often the data is sampled. The larger the window, the rougher the estimate and the wider the resulting envelop or thresholds will have to be. The smaller the window, the more accurate the approximation. However, with a smaller window comes the need to collect and process more data,

## Measurement Baselines



with resulting load on the network and measurement server.  
Configurable parameter in the Service Model (default is one hour).

**Figure 3. - Measurement Baselines and Thresholds**

---

**Example**

Let's say that you want to measure POP utilization at a specific POP in your network. This measurement is significant in many ways. If the utilization is too high, the POP is experiencing congestion, which results in longer access times, increased access failures, and lower customer satisfaction. On the other hand, if the utilization is too low it indicates that your customers are not using the service to the degree expected. This could be caused by something natural, but unexpected (for example, Super Bowl Sunday). However, it could also be caused by a failure in the access infrastructure (causing logins to be dropped or hang), or it could be due to customers logging off in frustration generated by some other problem in the system (web page response, email performance, and so forth).

Naturally, this measurement will have different values depending on whether it's 2:00 AM or 7:00 PM, and whether it's Monday or Sunday. Obviously a static threshold can't be used because the range of acceptable values depends on the time of day. If a fixed threshold is used, problems will either be left undetected or false alarms will become rampant.

To establish a baseline on this measurement, we start with the fixed Baseline Width (period) of one week. This compares measurements taken in a given time interval on a specific day of the week to measurements taken in the same interval in the preceding week. Next, a reasonable historical period or Baseline Length is set. In this case, let's say 8 weeks (the shorter the period, the lower the confidence factor). Finally, an Interval or window is established. To determine the window, you look at the volatility of the measurement data. A large slope around measurement peaks indicates that you need a small window. In this case you choose to change the default one-hour interval to 15-minutes to account for rapid fluctuations during the cycle.



Now that the baseline parameters have been configured, Firehunter automatically scans the measurement data over the most recent 8-week period, looking at individual 15-minute intervals over one day every week. Within each 15-minute interval a mean and standard deviation are calculated. The mean represents normal value of the measurement (baseline). The standard deviation represents the “envelope” of acceptable values and provides the basis for setting variable thresholds. Figure 4 shows an example of how baseline and threshold parameters are set in the Firehunter product.

**Edit Measurement Properties**

Properties | Data | Commands

ServerResponseTime

Full Name  
wehhost

Measurement Properties

Frequency: 5 minutes | Upload Frequency: 15 minutes

Baseline Properties

Smoothing Percentage: 0

Select a threshold

New\_threshold

New\_threshold

Type: ☒ Static ☐ Variable

Condition: ☒ Increasing ☐ Decreasing ☐ Envelope

Boundaries

<input checked="" type="radio"/>	Warning	84.134	Percentile
<input type="radio"/>	Minor	97.725	Percentile
<input type="radio"/>	Major	99.865	Percentile
<input type="radio"/>	Critical	99.997	Percentile

OK Cancel Apply Help

**Figure 4. – Setting Thresholds**

Multiple thresholds can be set. Variable thresholds are set as a percentile with default settings at one standard deviation for Warning increasing to four standard deviations for Critical. Thresholds can be either static or variable and can be set for increasing or decreasing

measurement crossings or mirrored positive/negative envelope thresholds around the baseline. Measurements crossing any threshold will trigger a corresponding event.

As Firehunter monitors the data, the 8-week window slides along. Thus, slow trends in usage, utilization, and so forth, are accounted for. Only abrupt changes from the norm trigger events. The trends are not ignored, however. The overall change (increase or decrease) of the curve provides valuable information for planning. This “baseline creep” can be used to determine when additional resources will be needed.

Firehunter will continue to calculate a baseline based on these parameters so that your most current data is included in the baseline continually. Slow trends in usage, utilization, and so on are accounted for. These trends can then be tracked by watching the “baseline creep.” The overall increase or decrease of the baseline curve can be used to determine when additional resources will be needed.

---

## Actions

By defining actions, you can instruct Firehunter to send e-mail or invoke a script when threshold or service health events occur.

You can specify that an action occur for specific measurements, for specific event types, and for specific severity levels. You can also indicate the action should only occur during specific time ranges.

This chapter describes possible actions you might want to take and provides examples that you can use to create your specific actions.

---

## Paging one or more people

You can instruct Firehunter to page someone by running a pager script. An example for this sort of script follows:

```
PAGE_RECIPIENT=$1
MESSAGE=$2
SEVERITY=$3
DO_CRITICAL=$4

if [ $SEVERITY = Critical -a $DO_CRITICAL = false ]
then
    exit 0
fi

invoke_pager "$PAGE_RECIPIENT" "$MESSAGE"
```

Perhaps when an event occurs you want to send e-mail to some people and page others depending on the time of day. An example for this sort of action follows:

Action Manager Editor										
File Configuration Help										
Boundary Health										
Active	Start	Stop	Notification Type	Notification	Server	Service	Measurement	Severity	Identific...	
<input checked="" type="checkbox"/>	12:00 AM	12:00 AM	Execute	bin\pager.bat "kelly joe jeni" "\$M" \$3 true	ALL	ALL	ALL	Critical	critical...	
<input checked="" type="checkbox"/>	8:00 AM	5:00 PM	Email	joe	ALL	ALL	ALL	ALL	shift1_...	
<input checked="" type="checkbox"/>	8:00 AM	5:00 PM	Execute	bin\pager.bat kelly "\$M" \$3 false	ALL	ALL	ALL	ALL	shift1_...	
<input checked="" type="checkbox"/>	5:00 PM	12:00 AM	Email	jeni	ALL	ALL	ALL	ALL	shift2_...	
<input checked="" type="checkbox"/>	5:00 PM	12:00 AM	Execute	bin\pager.bat joe "\$M" \$3 false	ALL	ALL	ALL	ALL	shift2_...	
<input checked="" type="checkbox"/>	12:00 AM	12:00 AM	Email	kelly	ALL	ALL	ALL	ALL	shift3_...	
<input checked="" type="checkbox"/>	12:00 AM	8:00 AM	Execute	bin\pager.bat jeni "\$M" \$3 false	ALL	ALL	ALL	ALL	shift3_...	

## Exporting a report to HTML

Perhaps when an event occurs you want to export a report to HTML so you could view relevant information from a browser. An example for this sort of action follows:

Action Manager Editor										
File Configuration Help										
Boundary Health										
Active	Notification Type	Notification	Node Name				Severity	Start	Stop	Identificati...
<input checked="" type="checkbox"/>	Execute	bin\report.bat -xsilent -n \$1 -d c:\firehunter	Web-Service:Http:lowrider:Http:TotalResponseTime				Critical	12:00 AM	12:00 AM	Web Report

## Sending an SNMP trap

Perhaps when an event occurs you want to send an SNMP trap to another network management tool. An example for this sort of action follows:

Action Manager Editor										
File Configuration Help										
Boundary Health										
Active	Notification Type	Notification	Node Name				Severity	Start	Stop	Identification
<input checked="" type="checkbox"/>	Execute	bin\snmptrap.bat HOST 0 "\$M"	Web-Service:Http:lowrider:Http:TotalResponseTime				Critical	12:00 AM	12:00 AM	Web Report

Integration with existing products

It's likely that you've already invested quite a bit of effort and/or money on other system monitoring tools. Firehunter doesn't squander that investment. Through the file system and command line execution, Firehunter integrates with your existing tools to leverage their capabilities and provide a consolidated view of your services. This chapter provides information about how to combine the power of Firehunter and your existing tools.

//reviewers, the sections have the ideas for the examples in them. What I need is 'real examples' to use to show how this was used by someone else and they are so happy yada yada. Can you help make these up or give real examples; might include graphs, etc.//

---

## Command line invocation

You can associate certain commands with each item in your service model, allowing you to correlate your model to the tools you already have. By assigning execution commands to the items in the service model, you can launch your tools against a certain element, passing parameters, such as the IP address, to the tool.

For example, you might want to open a telnet session to log into a particular server and reconfigure it.

---

## Data export

Most of the information gathered and monitored by Firehunter can be exported to either CSV or HTML files. You can then import this information into another tool for further analysis. For example, //need example, perhaps of using information in Excel?//



---

## Incorporation of measurements

As discussed in Chapter 4, you can add any measurements from your existing tools into Firehunter. This allows you to track all related measurements in the same program, and you can use these measurements to make comprehensive reports of your system's performance.

For example, perhaps you have network traffic metrics taking from a WAN Analyzer or WAN Probe that you would like to incorporate into Firehunter/PRO so you can perform SLA reporting on the network traffic levels. *//at what level should I stop? Do I need to tell how to do this?//*

---

## Actions

As discussed in Chapter 5, you might want to access one of your existing tools whenever a specific threshold is crossed. For example, you might want to launch your trouble ticketing software when certain critical thresholds are crossed.

---

## Service level query

Just as Firehunter can get information from your other tools and launch them as needed, you can also gather information from Firehunter as input to your other tools. For example, you might query Firehunter for information about specific measurement values over the previous 12 hours and feed them into a centralized database used for further analysis with data from other tools.

## SLAs and Reporting

Any information that you can view in Firehunter can be included in a report or exported to an HTML or a comma-separated file.

You might want to create a report to

- view the Service Level Agreements (SLAs) you have established with a particular customer to see how effectively you are meeting those agreements and when any violations may have occurred
- show the values of related measurements leading up to a threshold event
- provide accurate data for capacity planning analysis
- provide “report cards” to your customers of the service levels you are providing to them

You might export this information for reasons such as the following:

- To further analyze the data using additional tools. For example, you might export data to a comma-separated file and then bring it into a spreadsheet program to analyze trends for planning purposes.
- To provide browser access to Firehunter data. For example, you might want to regularly export selected reports to HTML for access by your help desk, business planning personnel, or your customers.
- To take a “picture in time” for historical or archival purposes. (Although you can view past data in Firehunter, data may be archived or removed after a certain period of time according to your site’s policies on data aging.)

This chapter describes the various reports you can create and export from Firehunter.

---

## Creating reports

You can create your own custom reports using any of the measurements in Firehunter, whether they are standard measurements or measurements you have added to Firehunter. These reports are created through the Planning View of the Firehunter Client GUI.

For example, you might want to create an email service report that includes POP3 and SMTP measurements on availability, response time, and email round trip time. Alternatively, you may choose to compare the throughput and delay through separate network paths. Reports can be generated for any measurement that is incorporated into your service model.

Reports can be generated as HTML or CSV (Comma-Separated Value) files. HTML output is a graphical representation of Planning Views generated with the Firehunter GUI. Command-line interfaces, with a rich set of options, also exist for configuring and generating reports at scheduled time periods.

Furthermore, these Firehunter reports can be generated directly through a web browser by interacting with a web server that resides on a Diagnostic Measurement Server.

---

## SLA reports

To remain competitive in the industry, you have to offer availability, reliability and performance guarantees to your customers. Obviously, you don't want to make agreements you can't keep. Firehunter's SLA capabilities enable you to

- analyze prospective SLAs to evaluate various Service Level Objectives (SLOs) and demonstrate that they can be met before you offer them to your customers.

- monitor the quality of service provided to ensure you are meeting your contracts
- generate reports of varying detail, tuned for different audiences that capture your SLA results for specified times
- use the results to set thresholds that are tighter than those promised to your customers so that you are alerted to potential problems long before the customer experiences service degradation.

### SLA Examples

Any measurement that has been integrated into or collected by Firehunter can be incorporated into an SLA. Firehunter provides a rich set of specification options to characterize the SLAs you establish with your customers. Typically, these SLAs are based on customer support, performance, and reliability metrics. Some examples include:

- Customer Support--These include the typical help desk problem reporting and problem resolution guarantees. For example, you may offer 24x7 support, a single point of contact assigned to the customer, and problem resolution within 48 hours of reporting, and your SLA may represent your responsiveness when and if problems occur.
- Reliability--Reliability metrics consist of availability guarantees over a period of time. For example, you may promise no more than 20 minutes of unscheduled server downtime during the month and that the web server will be available 99.9% of the time it is accessed during the work week.
- Response time--Such metrics define the time a service request is allowed to take to respond to user requests. An example of this metric may be: Users, on the average, will not experience response time greater than 3 seconds for their requests over any consecutive two minute period.
- Throughput--This metric defines the rate at which data is delivered to the customer. An example could be: VPN users will be able to

download a 100Kb GIF file in under 5 seconds during working hours.

- **Utilization**—Such a metric measures the degree to which resources are being used in delivering your services. An example of this metric could be: the host system will support 64 simultaneous users within specified performance parameters, during peak business hours.

### **Assembling SLAs in Firehunter**

The fundamental component in SLAs is the **Service Level Objective (SLO)**. An SLA is composed of one or more SLOs, and each SLO has three parts to it:

1. **Measurement.** Any measurement in Firehunter can be used as a basis for an SLO. The SLO characterizes the criteria which determine when a measurement violates an agreement.
2. **Condition.** The condition is the criteria against which the measurement is compared to specify when and if the SLO is violated. Conditions can be set against static or variable boundaries; variable conditions are compared against baseline statistics which Firehunter collects to characterize "normal" (historical) behavior for the measurement at various times of the day for each day of the week. Envelope conditions may also be specified.

Conditions also incorporate the concept of a **Grace Period**. The grace period is a consecutive interval of time over which the condition must be consistently violated before the SLO itself is considered to be in violation. This is an important aspect of SLAs as applied to Internet services since much of the traffic over the Internet is bursty in nature. This burstiness may lead to short-lived, intermittent periods where a condition is in violation, but nevertheless tolerable to users due to their short-lived duration. A grace period ensures that only *sustained* Condition violations result in true SLO violations.

3. **Operating Time.** The operating time specifies when the SLO is in effect and should be evaluated. Specific calendar dates can be set, as well as given days of the week and given hours of the day.

Once SLOs are established, they are logically combined through **SLA Expressions**. SLA Expressions are defined to combine multiple SLOs through logical AND, OR and NOT expressions.

Note that different SLOs may be based on the same measurement. This allows the SLA to specify different conditions for a measurement at different operating times (e.g. one level of quality during the week, and another for weekends.)

Once all the the SLOs of interest have been combined into SLA Expressions, they are brought together into overall SLA definition. Some of the key attributes of the SLA include the **Contract Date**, which is when the SLA went into effect, and the Conformance Period. The Conformance Period defines the period of time over which an SLA is evaluated to determine whether it is in compliance or not. Example conformance periods are: 1 Month, 5 Days, 12 hours, 2 weeks, etc.

### Generating SLA reports

SLA reports can be generated either through its command-line interface or through a web browser aimed at the web server on a Diagnostic Measurement Server. Reports can be configured to include or exclude different amounts of information, depending on the intended audience. Content, which can be customized, includes:

- Contract and customer information
- The SLO specifications, with conditions and operating times, that make up the SLA
- A summary of overall compliance
- Specified times and corresponding data for when the SLA was violated
- Detailed measurement data that was analyzed in evaluating the SLA

SLA reports can be generated for any number of historically completed Conformance Periods. Additionally, the Conformance Period currently underway, but not yet completed, can be reported on to assess whether the SLA is currently being met. Also, an SLA report may be applied to more than one SLA, if desired.

You can use these reports in your discussions with your customers, to refine your SLAs or to provide periodic status reports demonstrating your ability to meet these agreements.



---

# Appendix A:

## License and Warranty

### **THE SOFTWARE LICENSE AGREEMENT**

ATTENTION: THIS SOFTWARE IS COPYRIGHTED BY HEWLETT-PACKARD COMPANY, 1998. ALL RIGHTS RESERVED.

IN ADDITION THE USE OF THE SOFTWARE IS SUBJECT TO THE HP  
SOFTWARE LICENSE TERMS SET FORTH BELOW.



ATTENTION: USE OF THE SOFTWARE IS SUBJECT TO THE HP SOFTWARE LICENSE TERMS SET FORTH BELOW. USING THE SOFTWARE INDICATES YOUR ACCEPTANCE OF THESE LICENSE TERMS. IF YOU HAVE PURCHASED THIS PRODUCT AND DO NOT ACCEPT THESE LICENSE TERMS, YOU MAY RETURN THE SOFTWARE FOR A FULL REFUND. IF THE SOFTWARE IS BUNDLED WITH ANOTHER PRODUCT, YOU MAY RETURN THE ENTIRE UNUSED PRODUCT FOR A FULL REFUND.

## **HP SOFTWARE LICENSE TERMS**

The following License Terms govern your use of the accompanying Software unless you have a separate written agreement with HP.

**License Grant.** "Use" means storing, loading, installing, executing or displaying the Software. For all portions of the Software, other than those portions that HP designates as "Agent Software" or "Remote GUI Software", HP grants you a license to Use the Software set forth in your Entitlement Certificate on a single designated processor to manage the number of network service elements set forth in the entitlement certificate. For those portions of the Software that HP designates as "Agent Software" HP grants you a license to Use multiple copies of the Agent Software. You may not modify the Software or disable any licensing or control features of the Software.

**Evaluation License:** If the software has been provided for evaluation or trial purposes then, notwithstanding any other provision in this agreement; A) your use of software is limited to evaluating the software for the period specified by HP and for the sole purpose of deciding whether to purchase a license to the software and B) the software is provided "AS IS" and HP will not be liable for any damages whatsoever.

**Ownership.** The Software is owned and copyrighted by HP or its third party suppliers. Your license confers no title to, or ownership in, the Software and is not a sale of any rights in the Software. HP's third party suppliers may protect their rights in the event of any violation of these License Terms.

**Copies and Adaptations.** You may only make copies or adaptations of the Software for archival purposes or when copying or adaptation is an essential step in the authorized Use of the Software. You must reproduce all copyright notices in the original Software on all copies or adaptations. You may not copy the Software onto any public network.

**No Disassembly or Decryption.** You may not disassemble or decompile the Software unless HP's prior written consent is obtained. In some jurisdictions, HP's consent may not be required for limited disassembly or decompilation. Upon request, you will provide HP with reasonably detailed information regarding any disassembly or decompilation. You may not decrypt the Software unless decryption is a necessary part of the operation of the Software.

**Transfer.** Your license will automatically terminate upon any transfer of the Software. Upon transfer, you must deliver the Software, including any copies and related documentation, to the transferee. The transferee must accept these License Terms as a condition to the transfer.

**Termination.** HP may terminate your license upon notice for failure to comply with any of these License Terms. Upon termination, you must immediately destroy the Software, together with all copies, adaptations and merged portions in any form.

**Export Requirements.** You may not export or re-export the Software or any copy or adaptation in violation of any applicable laws or regulations.

**U.S. Government Restricted Rights.** The Software and any accompanying documentation have been developed entirely at private expense. They are delivered and licensed as "commercial computer software" as defined in DFARS 252.227-7013 (Oct 1988), DFARS 252.211-7015 (May 1991) or DFARS 252.227-7014 (Jun 1995), as a "commercial item" as defined in FAR 2.101(a), or as "Restricted computer software" as defined in FAR 52.227-19 (Jun 1987)(or any equivalent agency regulation or contract clause), whichever is applicable. You have only those rights provided for such Software and any accompanying documentation by the applicable FAR or DFARS clause or the HP standard software agreement for the product involved.

## **ADDITIONAL LICENSE TERMS APPLICABLE TO THE SUN JAVA PLATFORM INTERFACE**

**Java Platform Interface.** Licensee may not modify the Java Platform Interface ("JPI", identified as classes contained within the "java" package or any subpackages of the "java" package), by creating additional classes within the JPI or otherwise causing the addition to or modification of the classes in the JPI. In the event that Licensee creates any Java-related API and distributes such API to others for applet or application development, Licensee must promptly publish broadly, an accurate specification for such API for free use by all developers of Java-based software.

**Restrictions.** Software is confidential copyrighted information of Sun and title to all copies is retained by Sun and/or its licensors. Licensee shall not decompile, disassemble, decrypt, extract, or otherwise reverse engineer Software. Software may not be leased, assigned, or sublicensed, in whole or in part, except as specifically authorized in Section 1. Software is not designed or intended for use in online control of aircraft, air traffic, aircraft navigation or aircraft communications; or in the design, construction, operation or maintenance of any nuclear facility. Licensee warrants that it will not use or redistribute the Software for such purposes.

**Trademarks and Logos.** This License does not authorize Licensee to use any Sun name, trademark or logo. Licensee acknowledges that Sun owns the Java trademark and all Java-related trademarks, logos and icons including the Coffee Cup and Duke ("Java Marks") and agrees to: (i) comply with the Java Trademark Guidelines at <http://java.sun.com/trademarks.html>; (ii) not do anything harmful to or inconsistent with Sun's rights in the Java Marks; and (iii) assist Sun in protecting those rights, including assigning to Sun any rights acquired by Licensee in any Java Mark.

**Disclaimer of Warranty.** Software is provided "AS IS," without a warranty of any kind. ALL EXPRESS OR IMPLIED REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED.

**Limitation of Liability.** SUN AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE OR ANY THIRD PARTY AS A RESULT OF USING OR DISTRIBUTING SOFTWARE. IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**ADDITIONAL LICENSE TERMS APPLICABLE TO THE AdventNet Java SNMP Package**

Copyright (c) 1996-98 AdventNet, Inc. All Rights Reserved.

Permission to use, copy, and distribute this compiled binary software and its documentation, except the Application Programming Interface documentation, in unmodified form without fee is hereby granted provided that this copyright notice appears in all copies. This software may not be distributed in any modified form or distributed with the API programming documentation without prior consent from AdventNet, Inc. Incorporating this software in any development tool such as a compiler, builder or an API requires a separate License from AdventNet, Inc.

Portions of this software were derived from the CMP SNMP 1.2U distribution and the following notice applies to the CMU software.

Copyright 1989 by Carnegie Mellon University. All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of CMU not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

**ADDITIONAL LICENSE TERMS APPLICABLE TO THE GifEncoder**

Copyright (C)1996,1998 by Jef Poskanzer <jef@acme.com>. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:



1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.



2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.